

10/05/00  
JC929 U.S. PTO

10-10-00

A

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Date: October 5, 2000

File No. A-69621/DCA/SMF

**Box PATENT APPLICATION FEE**

Assistant Commissioner for Patents

Washington, DC 20231

"EXPRESS MAIL" MAILING LABEL NUMBER EL 683 947 445 US

DATE OF DEPOSIT October 5, 2000

I HEREBY CERTIFY THAT THIS PAPER OR FEE IS BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE "EXPRESS MAIL POST OFFICE TO ADDRESSEE" SERVICE UNDER 37 CFR 1.10 ON THE DATE INDICATED ABOVE AND IS ADDRESSED TO: BOX PATENT APPLICATION FEE, ASSISTANT COMMISSIONER FOR PATENTS, WASHINGTON, DC 20231.

TYPED NAME Kari Bateman

SIGNED

*[Signature]*

JC806 U.S. PTO  
09/680560  
10/05/00

Sir:

Transmitted herewith for filing is the patent application of Inventor(s): Burton A. HIPP

For: DYNAMIC SYMBOLIC LINK RESOLUTION

Enclosed are also:

Prior Art Statement.

☒ 5 Sheets of drawings, Formal     , Informal X

☒ An Assignment of the invention to: Ejasent Inc.

☒ Declaration and Power of Attorney for Patent Application.

☒ Verified Statement Claiming Small Entity Status-Small Business Concern

	(Col. 1) NO. FILED	(Col. 2) NO. EXTRA	SMALL ENTITY RATE FEE	OTHER THAN SMALL ENTITY RATE FEE
BASIC FEE			\$355	\$710
TOTAL CLAIMS	44 - 20	-24-	x 9 = \$216	x 18 = \$
INDEP CLAIMS	5 - 3	-2-	x 40 = \$80	x 80 = \$
MULTIPLE DEPENDENT CLAIM PRESENTED			+130 = \$	+260 = \$
If the difference in Col 1 is less than zero, enter "0" in Col. 2			TOTAL \$651	TOTAL \$

☒ Our Check No. 6331 in the amount of \$691.00 is enclosed to cover both the filing fee of \$651.00 and the assignment recordation fee of \$40.00.

☒ This application claims priority to Provisional Application Serial Nos. 60/157,727 filed October 5, 1999, 60/157,728 filed October 5, 1999, 60/157,729 filed October 5, 1999, 60/157,833 filed October 5, 1999, and 60/157,834 filed October 5, 1999.

☒ The Commissioner is hereby authorized to charge any additional fees which may be required, including extension fees, or credit any overpayment to Deposit Account No. 06-1300 (Order No. A-69621/DCA). Two copies of this sheet are enclosed.

Respectfully submitted,

*[Signature]*  
Steven M. Freeland, Reg. No. 42,555

FLEHR HOHBACH TEST ALBRITTON & HERBERT LLP

Four Embarcadero Center, Suite 3400

San Francisco, California 94111-4187

Telephone: (415) 781-1989

Fax: (415) 398-3249

1018269

(1.14) 08/95

OCT. 4. 2000 4:10PM

FLEHR HOHBACH TEST

NO. 9613 P. 28

Applicant or Patentee: Hipp  
Serial No.: Not AssignedAtty Docket No. A-69621/DCA/SMF  
Filed Herewith**VERIFIED STATEMENT (DECLARATION) CLAIMING SMALL ENTITY STATUS  
(37 CFR 1.9(f) and 1.27(c)) - SMALL BUSINESS CONCERN**

I hereby declare that I am an official of the small business concern empowered to act on behalf of the concern identified below:

NAME OF SMALL BUSINESS CONCERN: Ejasant Inc.  
ADDRESS OF SMALL BUSINESS CONCERN: 2490 Charleston Road  
Mountain View, CA 94043

I hereby declare that the above identified small business concern qualifies as a small business concern as defined in 13 CFR 121.12, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees to the United States Patent and Trademark Office, United States Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third party or parties controls or has the power to control both.

I hereby declare that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the invention entitled *Dynamic Symbolic Link Resolution* by inventors *Burton A. Hipp*, described in the specification filed herewith. If the rights held by the above identified small business concern are not exclusive, each individual, concern or organization having rights in the invention is listed below\* and no rights to the invention are held by any person, other than the inventor, who would not qualify as an independent inventor under 37 CFR 1.9(c) if that person made the invention, or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d), or a nonprofit organization under 37 CFR 1.9(e). \*NOTE: Separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27)

Name: \_\_\_\_\_

Address: \_\_\_\_\_

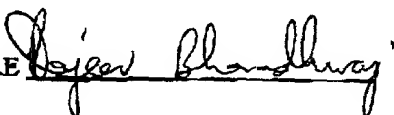
☐ Individual ☐ Small Business Concern ☐ Nonprofit Organization

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b)).

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

Name/Title: Rajeev Bharadhwaj, Founder & CTO  
Address: 2490 Charleston Road, Mountain View, CA 94043

SIGNATURE

DATE 10/4/00

1018171

## DYNAMIC SYMBOLIC LINK RESOLUTION

### REFERENCE TO RELATED APPLICATIONS

5  
10  
15  
The present application claims priority to and incorporates the following applications by reference: DYNAMIC SYMBOLIC LINK RESOLUTION, Prov. No. 60/157,728, filed on October 5, 1999; SNAPSHOT VIRTUAL TEMPLATING, Prov. No. 60/157,728, filed on October 5, 1999; SNAPSHOT RESTORE OF APPLICATION CHAINS AND APPLICATIONS, Prov. No. 60/157,833, filed on October 5, 1999; VIRTUAL RESOURCE-ID MAPPING, Prov. No. 60/157,727, filed on October 5, 1999; and VIRTUAL PORT MULTIPLEXING, Prov. No. 60/157,834, filed on October 5, 1999.

### FIELD

20  
25  
The present invention relates generally to resource management in a computer network. More specifically, the present invention relates to dynamic symbolic link resolution to discrete storage locations throughout a computer network.

### BACKGROUND

30  
35  
In prior art computer networks pointers or links are provided to indicate the location of a specific directory or file to allow a computer connected to the network to access or retrieve a specific file. Some prior art computational systems provide a mechanism for accessing a file pathname by alternate references. In this case, a particular file may have multiple pathnames that reference the file, with only one true path to the actual file or executable. These multiple references are commonly referred to as "shortcuts," aliases" or "symbolic links." Thus, prior art systems allow for a plurality of pathnames to access the same file. However, there is no function or mechanism for allowing a single pathname to access a plurality of different directories or files.

## SUMMARY

To achieve the foregoing, and in accordance with the purpose of the present invention, a system or network is disclosed which provides for a dynamic symbolic link (DSL) and the resolution of that DSL. The invention provides a method and apparatus that renames a first  
5 pathname to a target pathname, determines a variable within the target pathname, defines the first pathname as a symbolic link and associates the symbolic link with a virtual pathname. The present invention further defines a specification associated with the virtual pathname including associating the variable with the virtual pathname. In associating the symbolic link with the virtual pathname, the present invention further define a declaration within the virtual  
10 pathname.

## BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

FIG. 1A is a high level block diagram illustrating the various components of a computer network used in connection with the present invention;

FIG. 1B is a high level block diagram depicting a computer used in connection with the present invention;

FIG. 2 depicts a simplified block diagram for creating a Dynamic Symbolic Link (DSL);

FIG. 3 shows, in accordance with another aspect of the present invention, a simplified block diagram of the registration of a DSL specification;

FIG. 4 shows, in accordance with another aspect of the present invention, a simplified block diagram of the resolution of the DSL; and

FIG. 5 shows, in accordance with another aspect of the present invention, a simplified flowchart of the resolution of a DSL.

## DETAILED DESCRIPTION

Among other aspects and innovations, the invention provides structure, system, method, method of operation, computer program product, and business model and method for providing dynamic symbolic links and resolution of those dynamic symbolic links. The inventive dynamic symbolic link and resolution provided by the present invention inverts the  
35 paradigm of the prior linking mechanisms. The present dynamic symbolic link and resolution enables a computer network, computer, processor, microprocessor, server and other

computational systems to access a plurality of different directories, files or executables through a single dynamic symbolic link.

Where the prior art symbolic links allow a plurality of different links to access a single file, the present novel Dynamic Symbolic Link (DSL) and resolution of that DSL allows a computational system implementing the DSL and DSL resolution to access a plurality of different files, directories or executables from a single DSL.

By allowing access to a plurality of different directories or files from a single DSL, the present invention, in one embodiment, allows a plurality of instances of the same application to operate on a single computer without the applications interfering with one another. A first instance of an application will access, alter and/or store files, executables and data utilizing the DSL as resolved for the first instance. A second instance of an application will access, alter and/or store files, executables and data utilizing the DSL as resolved for the second instance, and so on. This isolates the files and data for each application instance, and thus prevents interference between instances of the application. Each instance of the application is capable of independently utilizing the computational resources without the need to reconfigure the application or the computational resources, thus the application is not limited to a specific computer.

In one embodiment, the present invention facilitates applications, such as Web servers and database applications, to be configured to operate in a computational network or system which includes a plurality of computers such that the applications are capable of being operated on a plurality of the computers within the system without reconfiguring the application or the computer. An application can be halted or frozen during operation on a first computer and reinitiated on a second computer without interrupting the application and without the need to reconfigure the second computer or application.

In one embodiment, isolating the instances of an application, and avoiding the necessity of limiting an instance to a specific computer network, allows the instance to be halted and reinitialized or reactivated on a second, different computer network. This frees up resources of the first computer network for other applications or instances.

When an application or process is operating, the application utilizes file pathnames and directory pathnames to access specifically defined or allocated portions of memory storage, for example, "/temp." However, if two or more instances of the same application attempt to operate from the same computer, both instances will attempt to access the same specifically allocated portion of the memory storage "/temp." This results in inconsistencies, illogical states and often fatal errors. The present invention allows for two or more instances of the same application to operate from the same computer without interference. The virtualizing of pathnames allows each instance of the application to be directed or allocated to different portions of memory, for example, "/temp\_1" and "/temp\_2." Thus, two or more

separate instances of the same application can operate independently on the same computer without interference.

Additionally, by allowing a specific instance of an application to access specific, predefined pathnames associated with that instance, the application and all accompanying data, operating states and associated directories and files can be isolated. Once isolated, the application and all accompanying data, states, directories and files can be stored for later use, as well as moved or transferred to operate on a different computer or network without affecting the operation of the application or other instances of the application. One advantage provided by this ability to store and transfer or reallocate an instance of an application is the optimization of computer and network resources. For example, if a first instance of an application is not currently in use, the computer can halt or freeze the application and all its accompanying states and data. The computer can then store the halted application and all accompanying states and data freeing up the computer to activate a second different application or second instance of the same application.

One example of the ability to halt or freeze an application along with all its accompanying states and data is described in co-pending U.S. Patent Application Ser. No. -- /- --, ---, entitled "Snapshot Virtual Templating," filed on October 5, 2000, incorporated herein by reference. One example of the ability to reinitialize an application along with all its accompanying states and data is described in co-pending U.S. Patent Application Ser. No. -- /- --, ---, entitled "Snapshot Restore Of Application Chains and Applications," filed on October 5, 2000, incorporated herein by reference.

The present invention provides snapshot virtual templating by creating virtual application templates for the purpose of propagating a single application snapshot into multiple, distinct images. Snapshot virtual templates allow multiple application instances to use the same fixed resource identifier by making the resource identifier virtual, privatizing it, and dynamically mapping it to a unique system resource identifier. When a snapshot is cloned from a virtual template, the common or shared data is used exactly as is, whereas the non-sharable data is either copied-on-write, multiplexed, virtualized, or customized-on-duplication. The present invention greatly reduces the required administrative setup per application instance. Snapshot virtual templating works by noting access to modified resources, fixed system IDs/keys and unique process-related identifies and automatically inserting a level of abstraction between these resources and the application. The resources contained in a snapshot virtual template can be dynamically redirected at restore time. Access to memory and storage is managed in a copy-on-write fashion. System resource handles are managed in a virtualize-on-allocate fashion or by a multiplex-on-access mechanism. Process-unique resources are managed in a redirect-on-duplicate fashion. Rules may be defined

through an application configurator that allows some degree of control over the creation of non-sharable data.

The present invention provides snapshot restoring by saving all process state, memory, and dependencies related to a software application to a snapshot image.

5 Interprocess communication (IPC) mechanisms such as shared memory and semaphores must be preserved in the snapshot image as well. IPC mechanisms include any resource that is shared between two process or any communication mechanism or channel that allow two processes to communicate or interoperate is a form of IPC. Sockets, shared memory, semaphores and pipes are some examples of IPC mechanisms. Between snapshots, memory

10 deltas are flushed to the snapshot image, so that only the modified-pages need be updated. Software modules that track usage of resources and their corresponding handles are included as part of the snapshot/restore framework of the present invention. At snapshot time, state is saved by querying the operating system kernel, the application snapshot/restore framework components, and the process management subsystem that allows applications to retrieve

15 internal process-specific information not available through existing system calls. At restore time, the reverse sequence of steps for the snapshot procedure is followed and state is restored by making requests to the kernel, the application snapshot/restore framework, and the process management subsystem.

20 The present method and system of providing DSL and DSL resolution allows a single dynamic symbolic link to be evaluated to multiple distinct values, retargeting the request transparently based upon application or process-specific configurations. In one embodiment, a DSL will have a declaration that contains a tag which will be matched with a DSL specification at the time of evaluation or resolution.

25 FIG. 1A illustrates in high level block diagram form the overall structure of the present invention as used in connection with a global computer network 100 such as the Internet. Remote users 102-1 and 102-2 can connect through the computer network 100 to a private network of computers 106 protected by firewall 104. Computer network 106 is a network comprising computers 108-1, 108-2, through 108-n, where n is the total number of

30 computers in network 106. Computers 150 are used to run various applications, as well as host web sites for access by remote users 102. The present invention is implemented on computer network 106 in the form of virtual environments 110-1 and 110-2. While only two virtual environments are illustrated, it is to be understood that any number of virtual environments may be utilized in connection with the present invention.

35 In one embodiment, the method and system of the present invention is implemented in a computer readable medium, such as a computer program 118 and executed on a computer 120 as illustrated in the high level block diagram of FIG. 1B. As shown, computer 120

incorporates a processor 122 utilizing, in one embodiment, a central processing unit (CPU) and supporting integrated circuitry. A memory 124 which is any type or combination of memory including fast semiconductor memory (e.g., RAM, NVRAM or ROM), slower magnetic memory (e.g., hard disk storage), optical memory and any conventional memory known in the art, to facilitate storage of the computer program 118 and the operating system software. In one embodiment, also included in computer 120 are interface devices including, but not limited to, keyboard 126, pointing device 130, and monitor 132, which allow a user to interact with computer 120. Mass storage devices such as disk drive 134 and CD ROM 136 may also be included in computer 120 to provide storage of information. Computer 120 may communicate with other computers and/or networks via modem 140 and telephone line 142 to allow for remote operation, or to utilize files stored at different locations. Other media may also be used in place of modem 140 and telephone line 142, such as a direct connection, high speed data line or a wireless connection, and the like. In one embodiment, the components described above may be operatively connected by a communications bus 144. In one embodiment, the components may be operatively connected by wireless communication.

In one embodiment, the DSL and DSL resolution system 150 provides a DSL which defines a variable pathname to a requested directory or file. In one embodiment, the DSL is incorporated or configured in alternative attribute fields in a file structure of a file system 236 (see FIG. 4). If a DSL is incorporated within a file attribute, a utility or application programming interface call is provided to set this attribute and designate the link as a DSL. In one embodiment, the DSL allows the system 150 to redirect the computer 120 to access one of a plurality of alternate directories or files as designated by the resolution of the DSL. One example of a DSL containing a DSL declaration is:

```
/home/${ID}/app/config (1)
```

In creating or associating the DSL, a file system 236 (see FIG. 4) entry is created which retargets the unique pathname of the file. In one embodiment, the retargeting is achieved by including at least one DSL declaration into the pathname to provide the DSL. In this embodiment, when the DSL declaration is included within a DSL, the DSL declaration is configured as a predefined, recognizable alphanumeric character sequence including a single alphanumeric character or an alphanumeric string. In example 1 above, the DSL declaration is defined as “\${ }.” In one embodiment, the DSL declaration includes a “tag.” The tag is a variable designation which is associated with one of a plurality of values. The tag is determined based on a DSL specification.

FIG. 2 shows a simplified block diagram of one implementation of one embodiment of the creation of a DSL. Target pathname 162 provides the destination location of an initial



or original file attempting to be accessed by an application. Initially, the original pathname 162 is evaluated to determine if multiple versions of the file designated by the pathname 162 are needed. This determination is based on predefined parameters. In one embodiment, if a pathname is determined to be utilized by more than one application, or more than one instance of the same application may be operated, and the information associated with the file at the pathname contains unique information for each application or instance of an application, then the DSL is created. In creating the DSL the original pathname 162, for example "/usr/app/config," is renamed or retargeted with a target file pathname 164, for example "/home/joe/app/config." The target pathname provides the path to the final storage location of the file 160. The target pathname 164 is then evaluated to determine which components 166 of the target pathname 164 are variable, for example "joe," to uniquely identify and distinguish the target pathname for the current instance of the application. The variable components of the target pathname, in one embodiment, are considered the tags. Once the variable components are determined, the DSL is generated replacing the variable components with the DSL declaration, for example "\${ID}," to be evaluated at run-time. The original file pathname 162 is then configured as a symbolic link 170. The symbolic link 170 is configured such that it points to the DSL 172, for example /home/\${ID}/app/config, to be resolved at run-time.

In creating and resolving the variable symbolic link 172, a DSL specification is utilized. In one embodiment, the DSL specification is associated with an individual process. When associating the tag with the DSL specification, the specification can be added explicitly by calling into an application programming interface (API), or specified implicitly. As an example, the DSL specification can be specified implicitly through an application environment, such as an in-memory application state that can be examined by the operating system. It may also be specified implicitly through an associated configuration mechanism, such as a registry or database containing parameters, properties and characteristics relevant to the particular application. The associated configuration mechanism is implemented, for example, as part of the operating system or as another discrete software component. For example, an application harness (which establishes the appropriate runtime environment for an associated application prior to the launching of the application) may make the call which sets the DSL specification on behalf of the application and then invokes the application which inherits the DSL specification.

In one embodiment, the DSL specification includes at least one symbolic tag and at least one value associated with that symbolic tag. Referring back to example 1 above, the DSL declaration is "\${ID}." The associated tag in this example is "ID." The DSL specification will include a reference to the symbolic tag "ID" as well as a value associated with the tag "ID".

FIG. 3 shows a block diagram of the one implementation of one embodiment of the registration of the DSL specification associated with an application 180 having a process<sub>x</sub> 182. When an application is initiated an application harness (including process<sub>x</sub>) 178 is signaled to launch. The application harness 178 registers (step 186) the DSL specification, including the tag 190 and associated value 192, with an operating system (OS) 194. The OS 194 records the DSL specification (step 196), and associates that DSL specification with the state of process<sub>x</sub> 210. The application 180 is then launched (step 212). In one embodiment, processes associated with the application 180 inherit the DSL specification of the process<sub>x</sub> of the application harness, such that when a process<sub>y</sub> 214 associated with application 180 is initiated, OS 194 assigns (step 216) the process<sub>y</sub> state 220 with the DSL specification from process<sub>x</sub> state 210. In one embodiment, processes associated with the application 180 have explicit DSL specifications and as such do not inherit the DSL specification.

In one embodiment, the DSL specification includes at least the symbolic tag and at least one associated value. For example, the DSL specification will provide a tag and value association similar to:

$$\begin{array}{cc} \text{tag} & \text{value} \\ \hline \text{ID} & \text{joe} \end{array} \quad (2)$$

As such, when the DSL declaration is identified, the DSL specification provides the associated value for the tag found within the DSL declaration. The value associated with the tag is substituted for the DSL declaration resulting in the target pathname. For example, the virtual pathname shown in example 1, using the DSL specification of example 2, will replace the DSL declaration “\${ID}” with the DSL specification value “joe,” resulting in an target pathname:

$$\text{/home/joe/app/config} \quad (3)$$

Once associated or resolved, the target pathname, for example “/home/joe/app/config” as shown in example 3, indicates the location of the target file. Because the DSL declaration can be resolved into a plurality of different values, for example joe, kim, dave, sue, x, temp, etc., a plurality of different pathnames can be resolved, each leading to different files (i.e., /home/kim/app/config; /home/dave/app/config; /home/sue/app/config; etc.), wherein each file is associated with a specific instance of an application.

FIG. 4 depicts a simplified block diagram of one implementation of one embodiment of the resolution of the DSL. An application 180 initiates a process, process<sub>y</sub>, which prompts (step 230) the OS 194 to access or open a directory or file, for example “/usr/app/config” 232. The OS 194 initiates a read (step 234) of the file and recognizes the pathname to be a symbolic link 170 in a file system 236. In one embodiment, file system 236 is any type or

combination of memory as described above. The file system 236 resolves the symbolic link and returns (step 238) a DSL 172 pointed to by the symbolic link 170. The OS 194 recognizes the instance of the DSL 172 by its DSL declaration and extracts the tag 190 from the DSL declaration. OS 194 then checks (step 240) process<sub>y</sub> for the DSL specification with a matching tag 190, for example "ID," associated with the process<sub>y</sub> state 220. Once the DSL specification is determined with the matching tag 190, the value 192, for example "joe," associated with the specific tag is returned (step 242). Once the value 192 is returned, the value 192 is substituted (step 244) into the DSL 172 resulting in the target pathname 164. The target pathname 164 is forwarded to the file system 236. The file system 236 will then return (step 246) a file handle 248, for example "/home/joe/app/config," to the application 180. In one embodiment, the file handle 248 is specifically associated with a single specific instance of an application, thus isolating the information from other applications.

Still referring to FIG. 4, in one embodiment, OS 194 includes an application snapshot (AppShot) framework 248. AppShot framework 248 provides routing for pathnames during the resolution of DSLs. In one embodiment, the AppShot framework is a software layer that resides between the application that can intercept API calls and track process state associated with an application. In one embodiment, requests for system resources or changes to process state are routed internally and the AppShot framework 248 tracks these events in anticipation of a snapshot.

In one embodiment, the application 180 initiating the DSL is unaware of the resolution of the DSL. The application 180 does not need to be modified to take advantage of the present DSL and DSL resolution. In one embodiment, a DSL specification is applied per process allowing a plurality of separate processes to obtain access to separate and distinct files which are referenced through a single pathname. Further, a process may have multiple DSL specifications. This allows a DSL specification to be created based on an intended use or functional use. For example, a first DSL specification may be used to redirect an application to the application's correct configuration file, while a second DSL specification may be used to redirect the application to the application's appropriate temporary file space area of memory.

In one embodiment, the method and apparatus allows the value associated with a tag to be designated or set as a default. When resolving the DSL, the OS 194 will search for the DSL specification with the matching tag 190. If the tag 190 is not found, the OS 194 will determine if a default value, for example "default," has been specified for the tag 190. If a default value has been set, then the OS 194 substitutes 244 the default value for the DSL declaration and accesses the file, for example, "/home/default/app/config," in file system 236. The default value is the value that is used in the absence of an explicit value. In one embodiment, if a dynamic symbolic link is encountered while referencing a file, and no DSL

specification with a matching tag has been set for that process, the default value may be utilized to resolve the pathname. For example, the DSL specification will provide a tag with a default value assigned similar to:

<u>tag</u>	<u>value</u>	
ID	default	(4).

A predefined alphanumeric character or string, such as "[:=" followed by one or more alphanumeric characters, such as "default," defining the default value, may be utilized to designate a default value if no other value is defined. In this example, the default is designated as "[:=default" with the value being equal to "default." In one embodiment, the default is derived from a character string contained within the DSL declaration. For example, if a DSL declaration is of the form "\${ID:=nobody}." Then "ID" is the tag portion of the DSL declaration, and "nobody" is defined as the default value for this declaration. If no match is found in resolving the tag "ID" (as described above), the default value "nobody" is used from the DSL declaration. The default is defined when the DSL declaration is created by adding a predefined alphanumeric character or string within the DSL declaration, for example "[:=" The predefined alphanumeric string is then followed by one or more alphanumeric characters defining the default value, such as "nobody."

In one embodiment, if DSL declaration is not explicitly defined and a default value has not been set, the tag is utilized as the default value. For example, if a DSL declaration is of the form "\${ID}," then "ID" is the tag portion of the DSL declaration, and "ID" is utilized as the value, resulting in a resolved path name such as "/home/ID/app/config." In one embodiment, if the DSL declaration is not explicitly defined and a default value has not been set, the DSL declaration, "\${ID}," is literally utilized as the default value for the declaration, resulting in a path name such as "/home/\${ID}/app/config," to access the file in the file system 236.

In one embodiment, the DSL generation and resolution of the present invention allows for a DSL to point to another DSL for further resolution. Thus, when OS 194 resolves a first DSL resulting in a first target pathname, the first target pathname points to a second DSL which will again be resolved by the OS 194 to a second target pathname or another DSL.

In one embodiment, a DSL may include a plurality of DSL declarations. Thus, upon resolution, the OS 194 resolves each of the DSL declarations the plurality of resolutions. For example, a DSL having a plurality of DSL declarations may be "/home/\${ID}/app/\${temp}." In one embodiment, the OS 194 would resolve one of the declarations, for example "\${ID}," then resolve the other, "\${temp}."

FIG. 5 depicts a flow diagram of an implementation of an embodiment of the resolution of a DSL according to method 258. In step 260, the application 180 or process 182

is initiated. In step 262, the application 180 or process 182 requests to open a file 232. In step 264, the pathname is analyzed to determine if the pathname is a symbolic link 170. If not, control transitions to step 266 where the requested file handle is returned. If, in step 264, the pathname is a symbolic link 170, control transitions to step 270, where the symbolic link 270 is resolved to a new pathname. In step 272, the new filename is analyzed to determine if it is a DSL. If not, control returns to step 262 where an attempt is made to open the file based on the new pathname. If, in step 272, the new pathname is a DSL, control transitions to step 274, where the tag 190 is extracted from the DSL declaration. In step 276, the DSL specification is determined, in one embodiment through a look-up table, based on the extracted tag 190. In step 280 it is determined if a match for the tag 190 was found. If not, the method returns to step 262 where the file based on the DSL filename is attempted to be opened. Alternatively, if the DSL specification is found in step 280, the method 158 transition to step 282 where the value 192 associated with the tag 190 is determined and substituted into the DSL for the DSL declaration. Control continues to step 284 where the DSL is further analyzed to determine if additional DSL declarations exist within the DSL. If there are additional DSL declarations or DSL tags, control transitions back to step 274 to extract the DSL tag 190. If, in step 284, there are no further DSL declarations or tags, control transitions to step 286 where the target pathname 246 is constructed by substituting the value associated with each tag into the pathname in place of the corresponding DSL declaration. Control then returns to step 262 where an attempt is made to open the target file 164.

As taught by the foregoing description and examples, a dynamic symbolic link and dynamic symbolic link resolution method and system are provided by the present invention. The foregoing description of specific embodiments and examples of the invention have been presented for the purpose of illustration and description, and although the invention has been illustrated by certain of the preceding examples, it is not to be construed as being limited thereby. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications, embodiments, and variations are possible in light of the above teaching. It is intended that the scope of the invention encompass the generic area as herein disclosed, and by the claims appended hereto and their equivalents.

Having disclosed exemplary embodiments and the best mode, modifications and variations may be made to the disclosed embodiments while remaining within the scope of the present invention as defined by the following claims.

WHAT IS CLAIMED IS:

1. A method for providing dynamic symbolic link resolution, comprising:  
receiving a first pathname;  
5 determining if the first pathname is a dynamic symbolic link (DSL);  
determining at least a first value associated with the DSL; and  
substituting the first value into the first pathname producing a first target pathname.
- 10 2. The method as claimed in claim 1, further comprising the step of:  
accessing a file designated by the first target pathname.
- 15 3. The method as claimed in claim 1, further comprising the step of:  
extracting at least one tag from the DSL; and  
utilizing the at least one tag in the step of determining at least the first value  
associated with the DSL.
- 20 4. The method as claimed in claim 3, wherein:  
the step of extracting the at least one tag including searching the DSL for at least one  
DSL declaration.
- 25 5. The method as claimed in claim 3, wherein:  
the step of extracting the at least one tag including searching the DSL for at least one  
predefined alphanumeric character sequence.
- 30 6. The method as claimed in claim 1, further comprising the step of:  
resolving the first pathname as a symbolic link to the DSL prior to the step of  
determining if the first pathname is the DSL.
7. The method as claimed in claim 1, further comprising the step of:  
returning a first file handle to the first target path.
8. The method as claimed in claim 1, wherein:  
the step of receiving the first pathname including receiving the first pathname from a  
first application attempting to access a first file;  
35 receiving the first pathname from a second application attempting to open the first  
file; and  
returning a second file handle to a second target pathname.

9. The method as claimed in claim 8, further comprising the step of:  
determining if the first pathname received from the second application is a DSL;  
determining at least a second value associated with the DSL; and  
substituting at least the second value into the first pathname producing the second  
5 target pathname following the step of receiving the first pathname from the second  
application.

10. The method as claimed in claim 9, wherein:  
the first application is a first instance of the first application; and  
10 the second application is a second instance of the first application.

11. The method as claimed in claim 1, further comprising the steps of:  
registering the DSL prior to the step of receiving the first pathname including:  
15 a) registering a DSL specification; and  
b) recording the DSL specification.

12. The method as claimed in claim 11, further comprising the steps of:  
receiving the DSL specification from a first process prior to the step of registering the  
DSL specification.

13. The method as claimed in claim 12, further comprising the steps of:  
the first application inheriting the DSL specification of the first process.

14. The method as claimed in claim 1, further comprising the steps of:  
generating the DSL prior to the step of receiving the first pathname including:  
25 a) obtaining the first pathname;  
b) renaming the first pathname as the first target pathname;  
c) determining at least one variable in the first target pathname; and  
d) creating a symbolic link from the first pathname pointing to the DSL.

15. The method as claimed in claim 14, further comprising the steps of:  
substituting the at least one variable in the first target pathname with a DSL  
declaration.

16. The method as claimed in claim 1, further comprising the steps of:  
determining a second value associated with the DSL; and

substituting the second value into the first pathname prior to producing the first target pathname.

17. The method as claimed in claim 1, further comprising the steps of:  
determining in a default value if the first value can not be determined; and  
substituting the default value into the first pathname producing the first target  
pathname.

18. The method as claimed in claim 1, wherein:  
the step of determining if the first pathname is a DSL including determining if the  
pathname includes at least one declaration;  
the step of determining the first value including:  
a) extracting a tag from the at least one declaration; and  
b) utilizing the tag as the first value; and  
the step of substituting the first value into the first pathname including substituting the  
tag as the first value into the pathname producing the first target pathname.

19. The method as claimed in claim 1, wherein:  
the step of determining if the first pathname is a DSL including determining if the  
pathname includes at least one declaration;  
the step of determining the first value including utilizing the declaration as the first  
value; and  
the step of substituting the first value into the first pathname including substituting the  
declaration as the first value into the pathname producing the first target pathname.

20. A computer system providing a method for providing a dynamic symbolic link,  
comprising the steps of:  
renaming a first pathname to a target pathname;  
determining a variable within the target pathname;  
defining the first pathname as a symbolic link; and  
associating the symbolic link with a virtual pathname.

21. The computer system as claimed in claim 20, further comprising the step of:  
defining a specification associated with the virtual pathname.



22. The computer system as claimed in claim 20, wherein:  
the step of defining a specification including associating the variable with the virtual  
pathname.

23. The computer system as claimed in claim 20, wherein:  
the step of associating the symbolic link including defining a declaration within the  
virtual pathname.

24. The computer system as claimed in claim 20, further comprising the steps of:  
resolving the dynamic symbolic link including:  
a) receiving a request for the symbolic link,  
b) determining the DSL;  
c) determining the target pathname from the DSL; and  
d) returning a handle to the target pathname.

25. The computer system as claimed in claim 24, wherein:  
the step of determining the target pathname including:  
a) extracting a DSL declaration from the DSL;  
b) extracting a tag from the DSL declaration;  
c) determining a value associated with the tag; and  
d) substituting the value into the DSL to generate the target pathname.

26. A method for enabling a dynamic pathname, comprising:  
receiving a request to access information stored under a pathname;  
determining if the pathname is dynamic;  
resolving the dynamic pathname; and  
providing access to the information stored under the resolved pathname.

27. The method as claimed in claim 26, wherein:  
the step of determining if the pathname is dynamic including determining if the  
pathname includes at least one declaration.

28. The method as claimed in claim 27, wherein:  
the step of resolving the dynamic pathname including:  
a) extracting a tag from the at least one declaration; and  
b) determining a value associated with the tag.

29. The method as claimed in claim 28, wherein:  
the step of providing access to the information including:  
a) substituting the value for the at least one declaration; and  
b) generating the resolved pathname.

30. The method as claimed in claim 27, wherein:  
the step of determining if the pathname is dynamic including determining if the  
pathname includes a plurality of declarations.

31. The method as claimed in claim 26, wherein:  
the step of resolving the dynamic pathname including:  
a) extracting a tag for each of the plurality of declarations;  
b) determining a value associated with each tag; and  
the step of providing access to the information including:  
a) substituting the associated value for each declaration; and  
b) generating the resolved pathname.

32. The method as claimed in claim 26, further comprising the steps of:  
creating the dynamic pathname including:  
a) determining a target pathname;  
b) determining at least one component of the target pathname to be dynamic;  
c) generating the dynamic pathname including replacing the at least one  
component with a declaration; and  
d) generating a symbolic link pointing to the dynamic pathname.

33. The method as claimed in claim 26, further comprising the steps of:  
defining a specification associated with the at least one component including:  
a) storing a tag; and  
b) storing a value associated with the tag.

34. A computer program product for providing access to stored information, the computer  
program product including a computer readable storage medium and a computer program  
mechanism embedded therein, the computer program mechanism comprising:  
a method of providing dynamic symbolic links (DSL) comprising:  
a) receiving a request to access information stored under a pathname;  
b) determining if the pathname is dynamic;  
c) resolving the dynamic pathname resulting in resolved pathname; and

d) providing access to the information stored under the resolved pathname.

35. The computer program product as claimed in claim 34, further comprising:  
the step of determining if the pathname is dynamic including

a) determining if the pathname includes at least one declaration.

36. The computer program product as claimed in claim 35, further comprising:  
the step of resolving the dynamic pathname including:

a) determining a first value associated with the at least one declaration; and

b) substituting the first value into the first pathname producing the resolved  
pathname.

37. The computer program product as claimed in claim 36, further comprising:  
the step of resolving the dynamic pathname including:

a) determining a default value if the first value is not determined; and

b) substituting the default value into the first pathname producing the resolved  
pathname.

38. The computer program product as claimed in claim 36, further comprising:  
the step of resolving the dynamic pathname including determining a tag from the  
declaration; and  
determining the first value associated with the tag.

39. The computer program product as claimed in claim 34, wherein:  
the step of providing access to the information including returning a file handle to the  
resolved pathname.

40. The computer program product as claimed in claim 34, further comprising:  
generating the DSL prior to the step of receiving the request to access the information  
stored under the pathname including:

a) obtaining the pathname;

b) renaming the pathname as the resolved pathname;

c) determining at least one variable in the resolved pathname;

d) substituting the at least one variable in the resolved pathname with a  
declaration; and

e) creating a symbolic link from the pathname pointing to the DSL.

41. The computer program product as claimed in claim 36, further comprising:  
registering the DSL prior to the step of receiving the request to access the information  
stored under the pathname including:

- a) receiving a DSL specification from an application;
- b) registering the DSL specification associated with the application; and
- c) recording the DSL specification.

42. The computer program product as claimed in claim 41, wherein:  
the step of receiving the DSL including receiving a tag and an associated value.

43. A computer network configured to provide dynamic symbolic link (DSL) and DSL  
resolution, the computational system comprising:

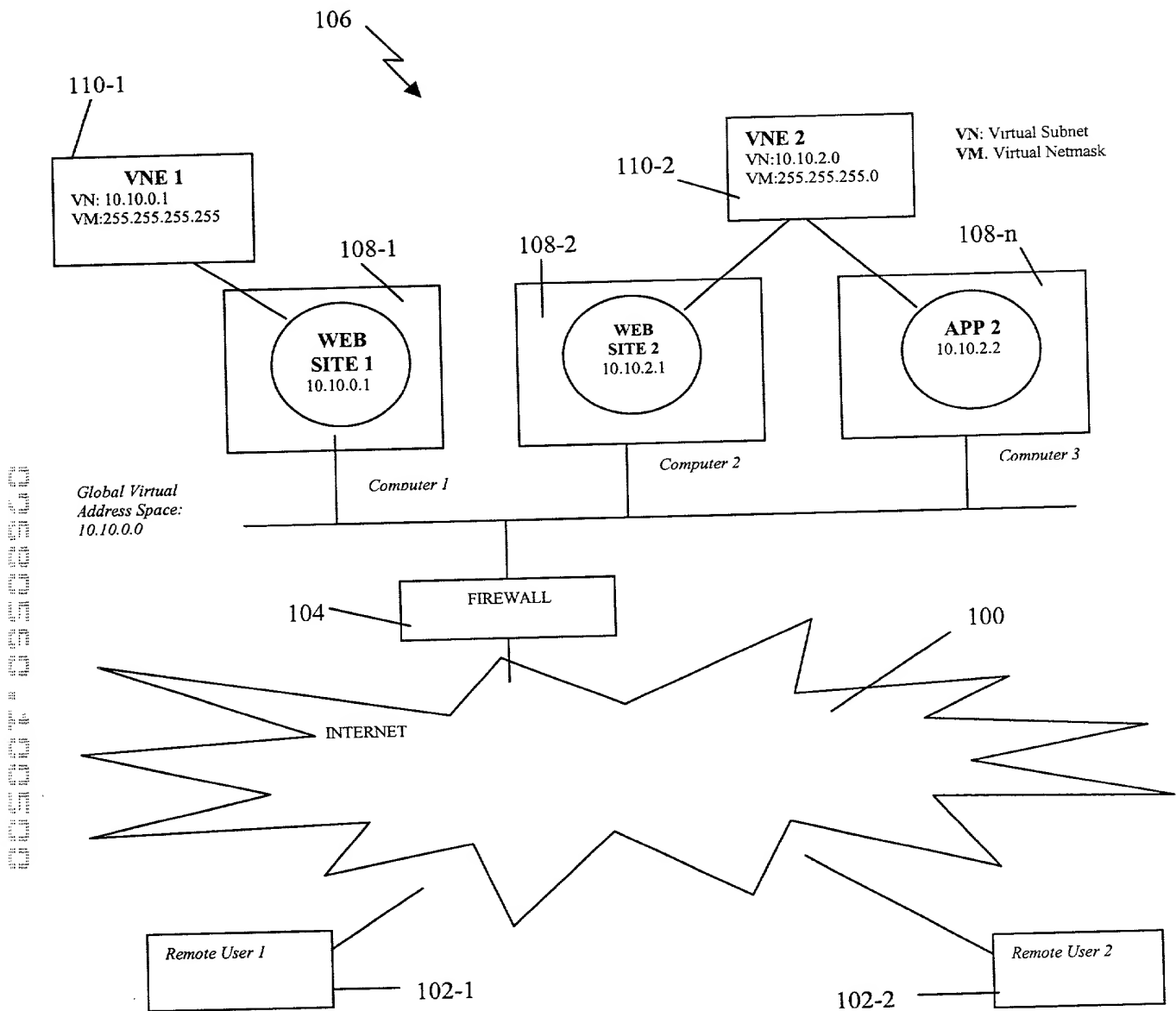
- a processor configured to run at least an operating system and at least one application;
- a memory coupled with the processor and configured to store information such that  
the information is stored and accessed through a pathname; and
- a means for accessing the stored information through the pathname including a means  
for dynamically resolving the pathname.

44. The computer network as claimed in claim 43, further comprising:  
the means for dynamically resolving the pathname including a means for determining  
if the pathname includes at least one declaration;

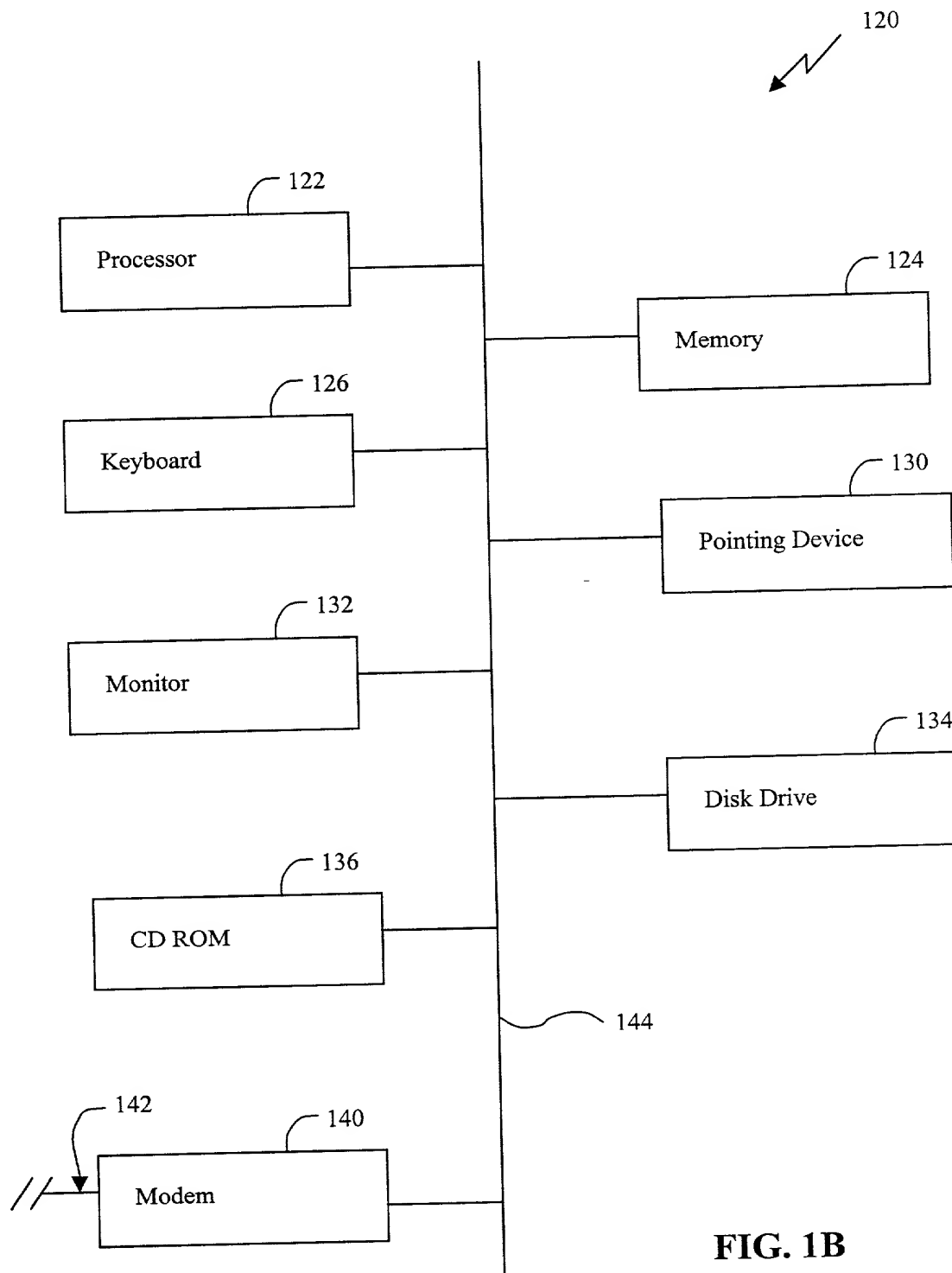
- a means for extracting a tag from the at least one declaration;
- a means for determining a value associated with the tag;
- a means for substituting into the pathname the value for the declaration providing a  
target pathname; and
- a means for accessing the target pathname including a means for returning a file  
handle of the target pathname to the application.

## ABSTRACT

To achieve the foregoing, and in accordance with the purpose of the present invention, a system or network is disclosed which provides for a dynamic symbolic link (DSL) and the resolution of that DSL. The invention provides a method and apparatus that renames a first  
5 pathname to a target pathname, determines a variable within the target pathname, defines the first pathname as a symbolic link and associates the symbolic link with a virtual pathname. The present invention further defines a specification associated with the virtual pathname including associating the variable with the virtual pathname. In associating the symbolic link  
10 with the virtual pathname, the present invention further define a declaration within the virtual pathname.



**FIG. 1A**



**FIG. 1B**

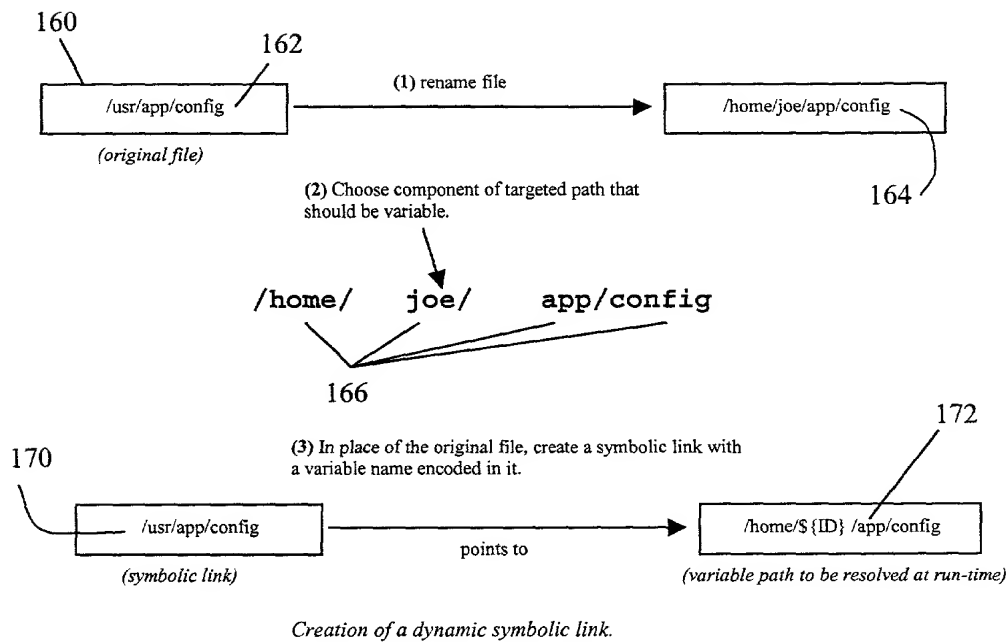
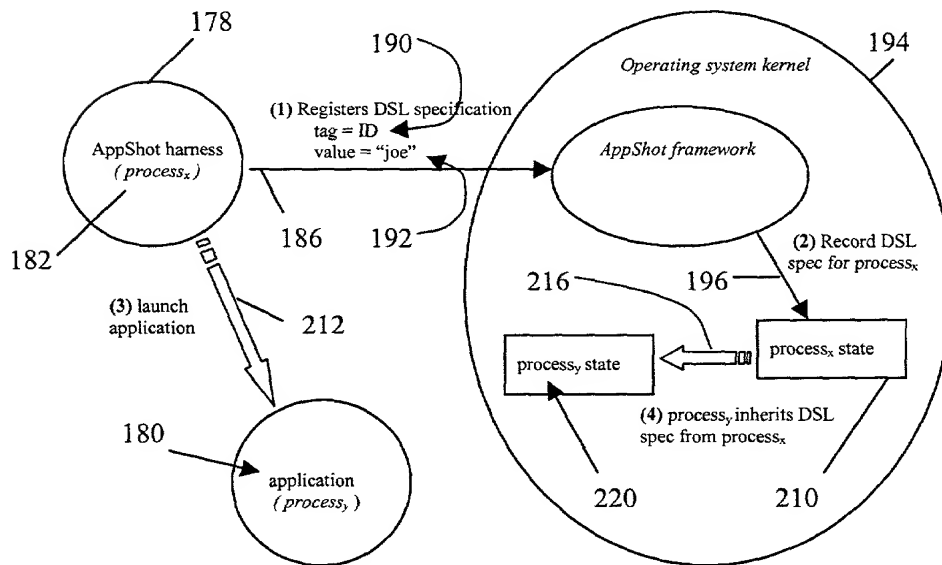


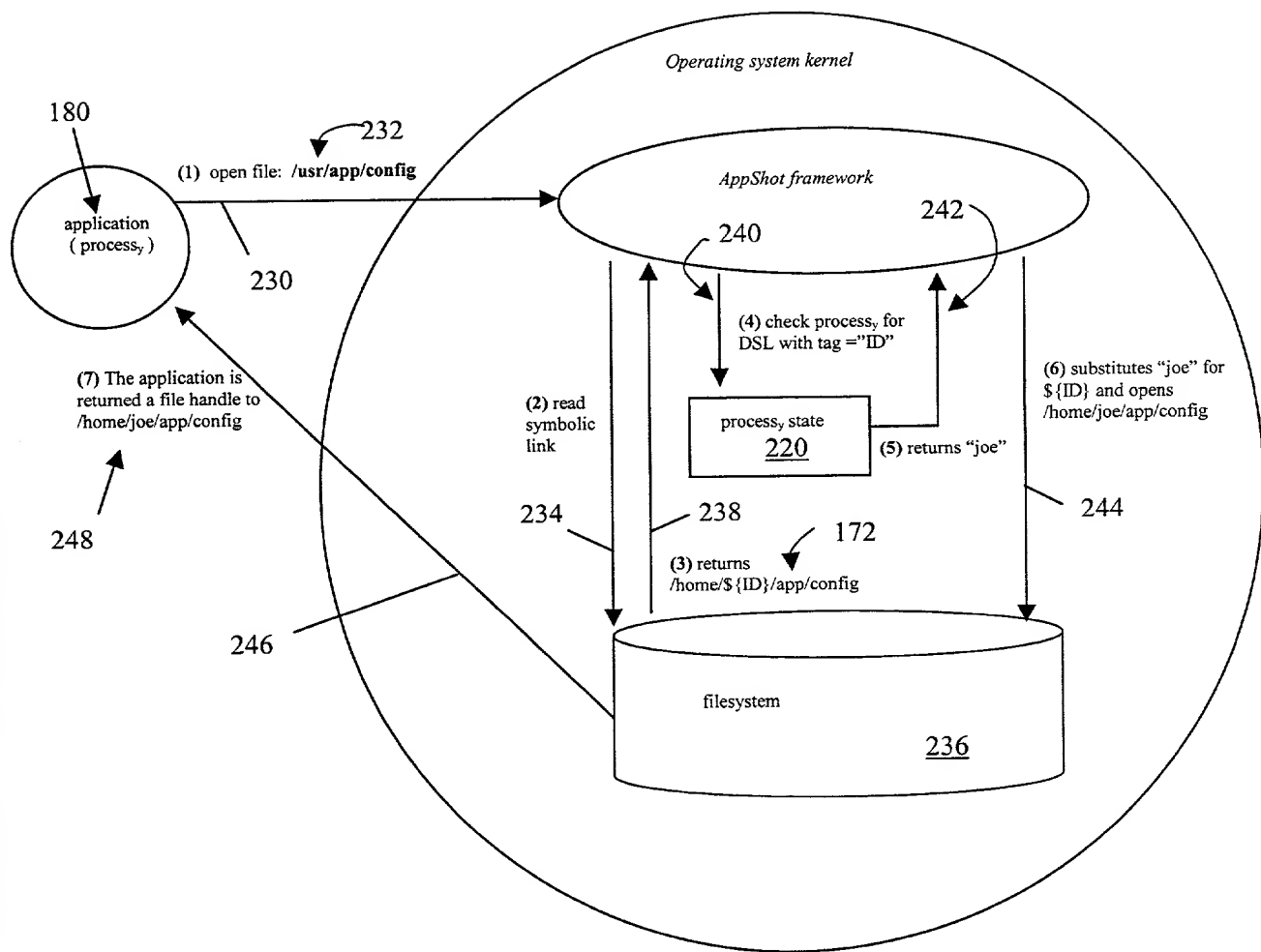
FIG. 2



Registration of a dynamic symbolic link specification.

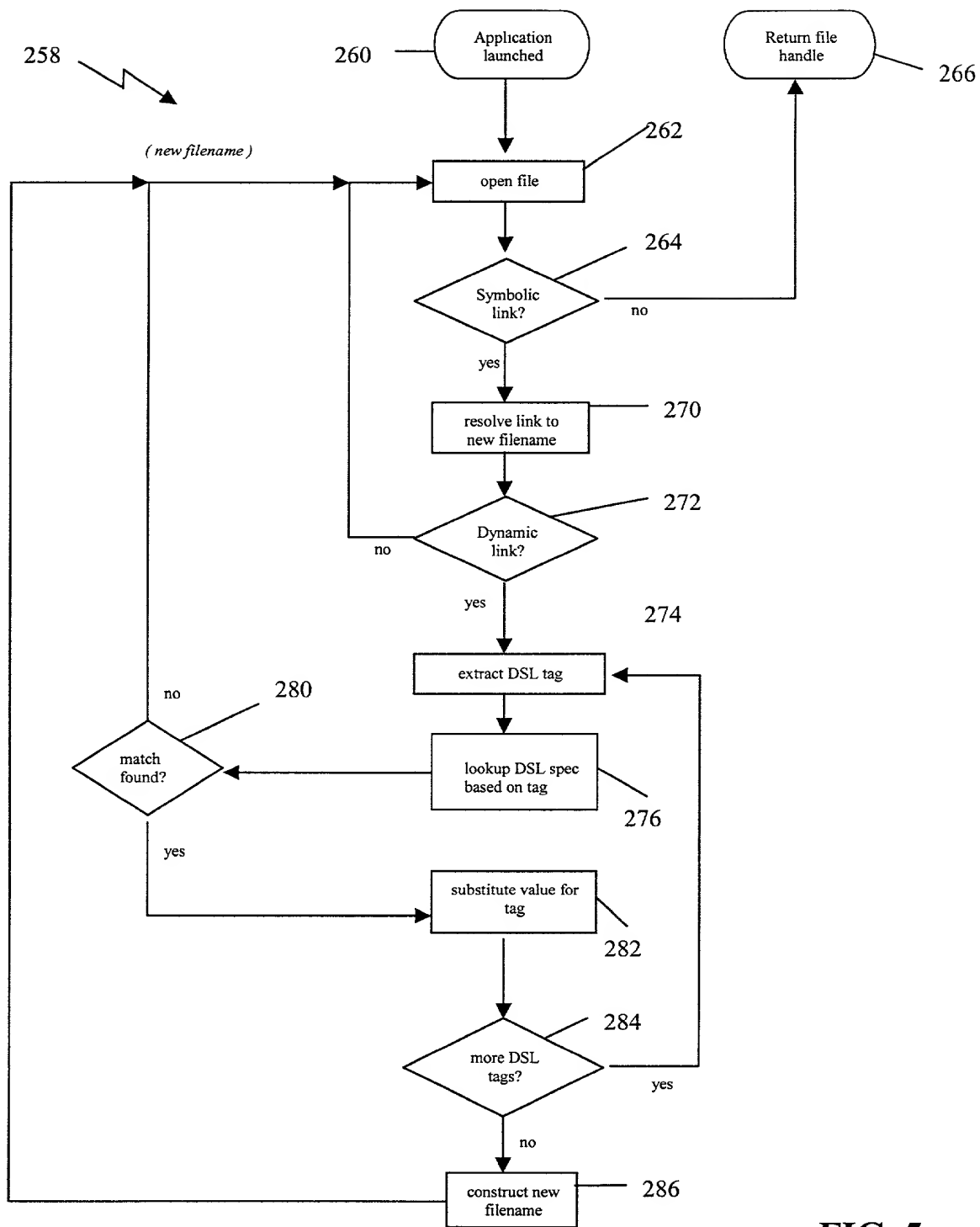
FIG. 3





Resolution of a dynamic symbolic link.

**FIG. 4**



Flowchart of dynamic symbolic link resolution.

FIG. 5

**DECLARATION AND POWER OF ATTORNEY  
FOR PATENT APPLICATION**

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name,

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled **DYNAMIC SYMBOLIC LINK RESOLUTION**, the specification of which

(check one) ☒ is attached hereto.  
☐ was filed on \_\_\_\_\_ as  
Application Serial No. \_\_\_\_\_  
and was amended on \_\_\_\_\_  
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose to the Patent Office all information known to me to be material to patentability as defined in 37 C.F.R. 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

_____	_____	_____
(Number)	(Country)	(Day/Month/Year Filed)
_____	_____	_____
(Number)	(Country)	(Day/Month/Year Filed)

Priority Claimed

<input type="checkbox"/>	<input type="checkbox"/>
Yes	No
<input type="checkbox"/>	<input type="checkbox"/>
Yes	No

I hereby claim the benefit under Title 35, United States Code, §119(e) of any United States provisional application(s) listed below.

<u>60/157,727</u>
(Application Serial No.)
<u>60/157,728</u>
(Application Serial No.)
<u>60/157,729</u>
(Application Serial No.)
<u>60/157,833</u>
(Application Serial No.)
<u>60/157,834</u>
(Application Serial No.)

<u>10/5/99</u>
(Filing Date)
<u>10/5/99</u>
(Filing Date)
<u>10/5/99</u>
(Filing Date)
<u>10/5/99</u>
(Filing Date)
<u>10/5/99</u>
(Filing Date)

<u>Pending</u>
(Status)
<u>Pending</u>
(Status)
<u>Pending</u>
(Status)
<u>Pending</u>
(Status)
<u>Pending</u>
(Status)

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose to the Patent Office all information known to me to be material to patentability as defined in 37 C.F.R. 1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

_____	_____	_____
(Application Serial No.)	(Filing Date)	(Status)

OCT. 4. 2000 4:10PM

FLEHR HOHBACH TEST

NO. 9613 P. 27

I hereby appoint the following attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith: Harold C. Hobbach, Reg. No. 17,757; Aldo J. Test, Reg. No. 18,048; Donald N. MacIntosh, Reg. No. 20,316; Edward S. Wright, Reg. No. 24,903; David J. Brezner, Reg. No. 24,774; Richard E. Backus, Reg. No. 22,701; James A. Sheridan, Reg. No. 25,433; Robert B. Chickering, Reg. No. 24,286; Richard F. Treccartin, Reg. No. 31,801; Steven F. Caserza, Reg. No. 29,780; Laura L. Kulhanjian, Reg. No. 33,257; Michael A. Kaufman, Reg. No. 32,988; Edward N. Bachand, Reg. No. 37,085; R. Michael Ananian, Reg. No. 35,050; Robin M. Silva, Reg. No. 38,304; David C. Ashby, Reg. No. 36,432; and Maria S. Swiatek, Reg. No. 37,244 : provided that if any one of said attorneys ceases being affiliated with the law firm of Flehr, Hobbach, Test, Albritton & Herbert as partner, employee or of counsel, such attorney's appointment as attorney and all powers derived therefrom shall terminate on the date such attorney ceases being so affiliated.

Direct all telephone calls to Steven M. Freeland at (650) 494-8700.

Address all correspondence to:

FLEHR HOHBACH TEST  
ALBRITTON & HERBERT LLP  
Suite 3400, Four Embarcadero Center  
San Francisco, California 94111

File No. A-69621/DCA/SMF

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Title 18, United States Code, §1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first

and sole inventor

Burton A. Hino

Inventor's signature:

[Signature]

Date:

10/4/00

Residence:

4117 E. Haack Ct., Elk Grove, CA 95758

Citizenship:

USA

Post Office Address:

Same as above